

[illegible]

3

Sy

MT

MT
MT

MT

MT

MT
MTMT
MTMT
MTMT
MT

MT
MT

MT
MT

MT

MT

MT
MT

MI
MT

MT
MT

MT
MT

MT
MT

MT
MT

MT

111

227

MT

MT
MT

MT

MT

M1
M1MT
MT

M1

M1

10

1

10

100

10

1

[illegible]

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL IIIII
LLLLLLLLLLL IIIII
SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```

MAC

(4)	44	Edit History
(5)	55	DECLARATIONS
(6)	92	MTH\$CVT D G - Convert D to G
(7)	199	CVT_COMMON - Common convert routine
(8)	233	CVT_D_G - Convert D to G
(9)	265	CVT_G_D - Convert G to D
(10)	322	CVT_HANDLER - Local condition handler

```
0000 1 .TITLE MTH$CVTDG - Convert D to G, G to D
0000 2 .IDENT /2-003/ ; File: MTHCVTDG.MAR Edit: JCW2003
0000 3
0000 4 :
0000 5 :*****
0000 6 :
0000 7 :*
0000 8 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 :* ALL RIGHTS RESERVED.
0000 11 :*
0000 12 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 :* TRANSFERRED.
0000 18 :*
0000 19 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 :* CORPORATION.
0000 22 :*
0000 23 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 :*
0000 26 :*****
0000 27 :
```



```

0000 29
0000 30 :++
0000 31 : FACILITY: Mathematics Library
0000 32 :
0000 33 : ABSTRACT:
0000 34 :
0000 35 :     Routines to convert a single value or a vector of values
0000 36 :     from D floating to G floating, and in reverse.
0000 37 :
0000 38 : ENVIRONMENT: User Mode, AST Reentrant
0000 39 :
0000 40 : --
0000 41 : AUTHOR: Steven B. Lionel, CREATION DATE: 23-Apr-79
0000 42 :

```

```

0000 44      .SBTTL Edit History
0000 45 :
0000 46 : 1-001 - Original.
0000 47 : 1-002 - Allow zero count address as omission. SBL 24-Apr-79
0000 48 : 1-003 - Fix bug in ZERO G. SBL 14-Nov-1979
0000 49 : 2-001 - Separate entry for array conversion. Fault on reserved operand.
0000 50 : SBL 31-Dec-1979
0000 51 : 2-002 - Use general mode addressing. SBL 30-Nov-1981
0000 52 : 2-003 - Removed an unnecessary .EXTRN SYSSUNWIND. JCW 19-JUN-84.
0000 53

```



```
0000 55      .SBTTL  DECLARATIONS
0000 56      :
0000 57      : INCLUDE FILES:
0000 58      :
0000 59      :
0000 60      :
0000 61      : EXTERNAL DECLARATIONS:
0000 62      :
0000 63      : .DSABL  GBL
0000 64      :
0000 65      :
0000 66      : .EXTRN  MTH$$SIGNAL
0000 67      : .EXTRN  MTH$K_FLOUNDMAT
0000 68      : .EXTRN  MTH$K_FLOOVEMAT
0000 69      :
0000 70      :
0000 71      : MACROS:
0000 72      :
0000 73      :   $CHFDEF
0000 74      :   $SSDEF
0000 75      :   $SFDEF
0000 76      :   $PSLDEF
0000 77      :
0000 78      : EQUATED SYMBOLS:
0000 79      :
0000 80      :
0000 81      :
0000 82      : OWN STORAGE:
0000 83      :
0000 84      :
0000 85      :
0000 86      : PSECT DECLARATIONS:
0000 87      :
00000000 88      : .PSECT  _MTH$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 89      : EXE, RD, NOWRT, LONG
0000 90
```

```
: Prevent undeclared
: symbols from being
: automatically global.
: Math signal routine
: Underflow error code
: Overflow error code
```

```
0000 92 .SBTTL MTH$CVT_D_G - Convert D to G
0000 93 :++
0000 94 : FUNCTIONAL DESCRIPTION:
0000 95 :
0000 96 : MTH$CVT_D_G and MTH$CVT_DA_GA convert D_floating values to
0000 97 : G_floating.
0000 98 :
0000 99 : MTH$CVT_G_D and MTH$CVT_GA_DA convert G_floating values to
0000 100 : D_floating.
0000 101 :
0000 102 : MTH$CVT_D_G and MTH$CVT_G_D are functions which convert their
0000 103 : single argument to the destination datatype and return it as
0000 104 : a function value.
0000 105 :
0000 106 : MTH$CVT_DA_GA and MTH$CVT_GA_DA are subroutines which convert
0000 107 : an array of values in a single call.
0000 108 :
0000 109 : These routines are designed to function like hardware convert
0000 110 : instructions. They will fault on reserved operands. If
0000 111 : overflow is detected, or underflow with FU enabled, an error
0000 112 : is signaled.
0000 113 :
0000 114 : All four routines are designed to function correctly on VAX-11
0000 115 : processors which do not have the G_floating instruction set.
0000 116 :
0000 117 : CALLING SEQUENCES:
0000 118 :
0000 119 : result.wg.v = MTH$CVT_D_G (source.rd.r)
0000 120 : result.wd.v = MTH$CVT_G_D (source.rg.r)
0000 121 :
0000 122 : CALL MTH$CVT_DA_GA (source.rd.ra, dest.rg.ra [, count.rl.r])
0000 123 : CALL MTH$CVT_GA_DA (source.rg.ra, dest.rd.ra [, count.rl.r])
0000 124 :
0000 125 : INPUT PARAMETERS:
0000 126 :
00000004 0000 127 : source = 4 ; Argument to be converted. Either
0000 128 : ; a scalar, if count is omitted, or
0000 129 : ; an array.
0000000C 0000 130 : count = 12 ; Optional. The count of array elements
0000 131 : ; in source and dest. If omitted, 1 is
0000 132 : ; assumed.
0000 133 :
0000 134 : IMPLICIT INPUTS:
0000 135 :
0000 136 : The callers PSL, which is examined to see if floating underflow
0000 137 : is enabled.
0000 138 :
0000 139 : OUTPUT PARAMETERS:
0000 140 :
0000 141 : value ; The converted value returned in R0-R1
0000 142 : ; if MTH$CVT_D_G or MTH$CVT_G_D.
00000008 0000 143 : dest = 8 ; The destination of the conversion. It
0000 144 : ; must be the same length as source.
0000 145 : ; If count is present, dest must also be
0000 146 : ; present.
0000 147 : ; Source and dest MUST either overlap
0000 148 : ; exactly, or be completely disjoint.
```



```
0000 149 :  
0000 150 : IMPLICIT OUTPUTS:  
0000 151 :  
0000 152 :     NONE  
0000 153 :  
0000 154 : FUNCTION VALUE:  
0000 155 :  
0000 156 :     If only source is given, the result is returned as the function  
0000 157 :     value in R0-R1.  
0000 158 :  
0000 159 : SIDE EFFECTS:  
0000 160 :  
0000 161 :     MTH$CVT_G_D signals MTH$_FLOOVEMAT (Floating overflow in Math  
0000 162 :     library) if conversion overflows result. Default result is  
0000 163 :     reserved operand.  
0000 164 :  
0000 165 :     MTH$CVT_G_D signals MTH$_FLOUNDMAT (Floating underflow in Math  
0000 166 :     library) if conversion underflows and the caller has floating  
0000 167 :     underflow enabled. The default result is zero.  
0000 168 :  
0000 169 :     All routines detect reserved floating operands by creating  
0000 170 :     a reserved operand fault (SS$_ROPRAND). If the reserved value  
0000 171 :     is changed to a non-reserved value, conversion will continue.  
0000 172 :  
0000 173 :  
00FC 0000 174 : .ENTRY MTH$CVT_DA_GA, ^M<R2,R3,R4,R5,R6,R7>  
0002 175 :  
54 0079'CF 9E 0002 176 : MOVAB W^CVT_D_G, R4 ; Address of actual convert routine  
25 11 0007 177 : BRB CVT_COMMON ; Join common routine  
0009 178 :  
0009 179 :  
00FC 0009 180 : .ENTRY MTH$CVT_GA_DA, ^M<R2,R3,R4,R5,R6,R7>  
000B 181 :  
6D 014A'CF 9E 000B 182 : MOVAB W^CVT_HANDLER, (FP) ; Enable local condition handler  
54 00C5'CF 9E 0010 183 : MOVAB W^CVT_G_D, R4 ; Address of actual convert routine  
17 11 0015 184 : BRB CVT_COMMON ; Join common routine  
0017 185 :  
0017 186 :  
00FC 0017 187 : .ENTRY MTH$CVT_D_G, ^M<R2,R3,R4,R5,R6,R7>  
0019 188 :  
54 0079'CF 9E 0019 189 : MOVAB W^CVT_D_G, R4 ; Address of actual convert routine  
3F 11 001E 190 : BRB FUNC_COMMON ; Join common routine  
0020 191 :  
0020 192 :  
00FC 0020 193 : .ENTRY MTH$CVT_G_D, ^M<R2,R3,R4,R5,R6,R7>  
0022 194 :  
6D 014A'CF 9E 0022 195 : MOVAB W^CVT_HANDLER, (FP) ; Enable local condition handler  
54 00C5'CF 9E 0027 196 : MOVAB W^CVT_G_D, R4 ; Address of actual convert routine  
31 11 002C 197 : BRB FUNC_COMMON ; Join common routine
```

```
002E 199 .SBTTL CVT_COMMON - Common convert routine
002E 200
002E 201 CVT_COMMON:
002E 202     MOVZBL #1, R7 ; Default count is 1
0031 203     MOVL source(AP), R5 ; Get source address
0035 204     MOVL dest(AP), R6 ; Get destination address
0039 205     CMPB (AP), #<count/4> ; Is count present?
003C 206     BLSS LOOP ; If not, use default of 1
003E 207     TSTL count(AP) ; Try other way
0041 208     BEQL LOOP ; Still not there
0043 209     MOVL @count(AP), R7 ; Get count
0047 210
0047 211 LOOP:
0047 212     ROTL #16, (R5)+, R1 ; Get operand and swap words
004B 213     ROTL #16, (R5)+, R0 ;
004F 214     JSB (R4) ; Do the appropriate conversion
0051 215     ROTL #16, R1, (R6)+ ; Store result
0055 216     ROTL #16, R0, (R6)+ ;
0059 217     SOBGTR R7, LOOP ; Loop until done
005C 218
005C 219     CLRQ R0 ; Just in case someone is looking
005E 220     RET ; Exit
005F 221
005F 222
005F 223 FUNC_COMMON:
005F 224     MOVL source(AP), R5 ; Get operand address
0063 225     ROTL #16, (R5)+, R1 ; Get single operand
0067 226     ROTL #16, (R5), R0 ;
006B 227     JSB (R4) ; Do the appropriate conversion
006D 228     ROTL #16, R0, R2 ; Swap words and longwords
0071 229     ROTL #16, R1, R0 ;
0075 230     MOVL R2, R1 ;
0078 231     RET ; Return
```



```
0079 233 .SBTTL CVT_D_G - Convert D to G
0079 234
0079 235 CVT_D_G:
52 51 08 17 EF 0079 236 EXTZV #23, #8, R1, R2 ; Get exponent in R2
1D 13 007E 237 BEQL ZERO_G ; If zero, return zero
53 50 01 02 EF 0080 238 EXTZV #2, #1, R0, R3 ; Save rounding bit
50 50 50 FD 8F 79 0085 239 ASHQ #-3, R0, R0 ; Shift right 3 bits
52 0380 8F A0 008A 240 ADDW2 #<1024-128>, R2 ; Change exponent bias from D to G
51 0B 14 52 F0 008F 241 INSV R2, #20, #11, R1 ; Insert G exponent
05 53 E9 0094 242 BLBC R3, EXIT_G ; Test for rounding
50 D6 0097 243 INCL R0 ; Round up
51 00 D8 0099 244 ADWC #0, R1 ; Propagate carry
009C 245
009C 246 EXIT_G:
05 009C 247 RSB ; Exit
009D 248
009D 249 ZERO_G:
0A 51 1F E0 009D 250 BBS #31, R1, RES_D ; Reserved operand?
50 D4 00A1 251 CLRL R0 ; Zero fraction and exponent
51 7FFFFFFF 8F CA 00A3 252 BICL2 #^X7FFFFFFF, R1 ; Leave sign alone
05 00AA 253 RSB ; Exit
00AB 254
00AB 255 RES_D:
52 50 10 9C 00AB 256 ROTL #16, R0, R2 ; Here if D floating reserved operand
50 51 10 9C 00AB 257 ROTL #16, R1, R0 ; Swap words, longwords
51 52 D0 00B3 258 MOVL R2, R1
50 50 73 00B6 259 TSTD R0 ; Will fault on reserved operand
52 50 10 9C 00B8 260 ROTL #16, R0, R2 ; Reswap and try again
50 51 10 9C 00BC 261 ROTL #16, R1, R0
51 52 D0 00C0 262 MOVL R2, R1
B4 11 00C3 263 BRB CVT_D_G
```

```
00C5 265 .SBTTL CVT_G_D - Convert G to D
00C5 266
00C5 267 CVT_G_D:
52 51 0B 14 EF 00C5 268 EXTZV #20, #11, R1, R2 ; Extract G exponent
22 13 00CA 269 BEQL ZERO_D ; Return zero if zero
53 51 01 1F EF 00CC 270 EXTZV #31, #1, R1, R3 ; Save sign
50 50 03 79 00D1 271 ASHQ #3, R0, R0 ; Shift left 3 bits
52 0380 8F A2 00D5 272 SUBW2 #<1024-128>, R2 ; Change bias from G to D
20 15 00DA 273 BLEQ UNDERFLOW ; Test for underflow
0100 8F 52 B1 00DC 274 CMPW R2, #256 ; Test for overflow
30 18 00E1 275 BGEQ OVERFLOW
51 08 17 52 F0 00E3 276 INSV R2, #23, #8, R1 ; Restore D exponent
51 01 1F 53 F0 00E8 277 INSV R3, #31, #1, R1 ; Restore sign
05 00ED 278 RSB
00EE
00EE
00EE
3C 51 1F E0 00EE 281 ZERO_D: BBS #31, R1, RES_G ; Reserved operand?
50 D4 00F2 282 CLRL R0 ; Zero fraction and exponent
51 7FFFFFFF 8F CA 00F4 283 BICL2 #^X7FFFFFFF, R1 ; Clear all but sign, so that
; reserved operand is set
; correctly.
05 00FB 284 RSB ; Return
00FB 285
00FC 286 UNDERFLOW:
00FC 287 MOVZWL SF$W_SAVE_PSW(FP), R2 ; Get caller's PSW
52 04 AD 3C 00FC 289 CLRQ R0 ; Default value is zero
50 7C 0100 290 BBC #PSL$V_FU, R2, ERROR_RET ; If not enabled, return zero
1C 52 06 E1 0102 291 MOVZBL #MTH$K_FLOUNDMAT, -(SP) ; Error code
7E 00 8F 9A 0106 292 CALLS #1, G^MTH$$SIGNAL ; Signal underflow
00000000 GF 01 FB 010A 293 BRB ERROR_RET ; Return
0F 11 0111 294
0113 295
0113 296 OVERFLOW:
50 01 0F 79 0113 297 ASHQ #15, #1, R0 ; Default reserved operand
7E 00 8F 9A 0117 298 MOVZBL #MTH$K_FLOOVEMAT, -(SP) ; Error code
00000000 GF 01 FB 011B 299 CALLS #1, G^MTH$$SIGNAL ; Signal overflow
0122 300
0122 301 ERROR_RET:
52 50 10 9C 0122 302 ROTL #16, R0, R2 ; Re-swap words, longwords
50 51 10 9C 0126 303 ROTL #16, R1, R0 ; in case we return with non-
51 52 D0 012A 304 MOVL R2, R1 ; zero value
05 012D 305 RSB ; Return
012E 306
012E 307 RES_G:
52 50 10 9C 012E 308 ROTL #16, R0, R2 ; Here if G floating reserved operand
50 51 10 9C 0132 309 ROTL #16, R1, R0 ; Swap words, longwords
51 52 D0 0136 310 MOVL R2, R1
50 53FD 0139 311 TSTG R0 ; Will fault on reserved operand
013C 312 ; If no G floating hardware,
013C 313 ; condition handler will take care
013C 314 ; of it.
013C 315 CVT_CONTINUE:
52 50 10 9C 013C 316 ROTL #16, R0, R2 ; Continue here from error
50 51 10 9C 0140 317 ROTL #16, R1, R0 ; Reswap and try again
51 52 D0 0144 318 MOVL R2, R1
FF7B 31 0147 319 BRW CVT_G_D
014A 320
```



```
014A 322 .SBTTL CVT_HANDLER - Local condition handler
014A 323
014A 324 :++
014A 325
014A 326
014A 327 CVT_HANDLER allows MTHSCVT_G_D and MTHSCVT_GA_DA to detect
014A 328 reserved operands using the TSTG instruction, regardless of
014A 329 whether the processor supports that instruction.
014A 330
014A 331 When CVT_G_D sees a reserved operand, it executes a TSTG with
014A 332 the G_floating operand in R0 and R1. If the processor knows
014A 333 about TSTG, a reserved operand fault is signaled. However,
014A 334 if it doesn't support TSTG, an "opcode reserved to Digital"
014A 335 fault will occur. CVT_HANDLER turns this into a reserved operand
014A 336 fault.
014A 337
014A 338 If the condition being signaled is not SSS_OPDEC or if the
014A 339 signaled instruction is not in the frame that established this
014A 340 handler, then the exception is resignaled. A test is made to
014A 341 see if the saved R0-R1 is a G_floating reserved operand. It
014A 342 will be on the initial fault, but might not be if it has been
014A 343 fixed up by another condition handler (i.e. LIB$FIXUP_FLT).
014A 344 If it is a reserved operand, the signal name is changed to
014A 345 SSS_ROPRAND and the exception is resignaled. Otherwise,
014A 346 execution continues with the instruction following the TSTG.
014A 347 :--
014A 348
014A 349 CVT_HANDLER:
014A 350 .WORD ^M<>
014A 351 MOVL 4(AP), R0 ; signal argument list
014A 352 CMPL CHF$S_SIG_NAME(R0), #SSS_OPDEC ; Opcode reserved to Digital fault?
014A 353 BNEQ RESIGNAL ; No, resignal
014A 354 MOVL 8(AP), R1 ; mechanism argument list
014A 355 TSTL CHF$S_MCH_DEPTH(R1) ; Is depth zero?
014A 356 BNEQ RESIGNAL ; If not, can't be this routine
014A 357 CMPZV #4, #12, CHF$S_MCH_SAVRO(R1), #^X800 ; Reserved operand?
014A 358 BNEQ CONTINUE ; No, continue with next instruction
014A 359 MOVL #SSS_ROPRAND, CHF$S_SIG_NAME(R0) ; Change condition code name
014A 360 RESIGNAL:
014A 361 MOVL #SSS_RESIGNAL, R0 ; Resignal exception
014A 362 RET
014A 363
014A 364 CONTINUE:
014A 365 SUBL3 #1, CHF$S_SIG_ARGS(R0), R1 ; Get position of PC
014A 366 MOVAL W^CVT_CONTINUE, (R0)[R1] ; Set return address
014A 367 MOVL #SSS_CONTINUE, R0 ; Continue execution
014A 368 RET
014A 369
014A 370 .END
```

0000043C 8F 04 AC 0000 014C 351
51 08 AC 0000 0150 352
08 A1 D5 015A 353
14 12 015E 354
00000800 8F 0C A1 0C 04 ED 0161 355
10 12 0163 356
04 A0 00000454 8F D0 016D 357
50 00000918 8F D0 016F 358
0177 359
0177 360
017E 361
017F 362
017F 363
51 60 01 C3 017F 364
6041 FFB5 CF DE 0183 365
50 01 D0 0189 366
04 018C 367
018D 368
018D 369
018D 370

MTHSCVTDG
Symbol table

- Convert D to G, G to D

N 12

16-SEP-1984 01:13:02
6-SEP-1984 11:21:31

VAX/VMS Macro V04-00
[MTHRTL.SRC]MTHCVTDG.MAR;1

Page 11
(10)

```

CHFSL_MCH_DEPTH      = 00000008
CHFSL_MCH_SAVRO      = 0000000C
CHFSL_SIG_ARGS       = 00000000
CHFSL_SIG_NAME       = 00000004
CONTINUE             = 0000017F R      02
COUNT               = 0000000C
CVT_COMMON           = 0000002E R      02
CVT_CONTINUE         = 0000013C R      02
CVT_D_G              = 00000079 R      02
CVT_G_D              = 000000C5 R      02
CVT_HANDLER          = 0000014A R      02
DEST                 = 00000008
ERROR_RET            = 00000122 R      02
EXIT_G               = 0000009C R      02
FUNC_COMMON          = 0000005F R      02
LOOP                 = 00000047 R      02
MTH$$SIGNAL          = ***** X      00
MTHSCVT_DA_GA        = 00000000 RG     02
MTHSCVT_D_G          = 00000017 RG     02
MTHSCVT_GA_DA        = 00000009 RG     02
MTHSCVT_G_D          = 00000020 RG     02
MTHSK_FLOOVEMAT      = ***** X      00
MTHSK_FLOUNDMAT      = ***** X      00
OVERFLOW             = 00000113 R      02
PSL$V_FU              = 00000006
RESIGNAL             = 00000177 R      02
RES_D                = 000000AB R      02
RES_G                = 0000012E R      02
SFS$ SAVE_PSW        = 00000004
SOURCE               = 00000004
SS$ CONTINUE         = 00000001
SS$ OPCDEC           = 0000043C
SS$ RESIGNAL         = 00000918
SS$ ROPRAND          = 00000454
UNDERFLOW            = 000000FC R      02
ZERO_D               = 000000EE R      02
ZERO_G               = 0000009D R      02

```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes															
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE					
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE					
_MTH\$CODE	0000018D (397.)	02 (2.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG					

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.11	00:00:01.02
Command processing	119	00:00:00.45	00:00:03.57
Pass 1	250	00:00:05.27	00:00:15.36

Symbol table sort	1	00:00:00.77	00:00:01.00
Pass 2	97	00:00:01.29	00:00:04.30
Symbol table output	7	00:00:00.07	00:00:00.20
Psect synopsis output	4	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	511	00:00:07.99	00:00:25.48

The working set limit was 900 pages.

29148 bytes (57 pages) of virtual memory were used to buffer the intermediate code.

There were 30 pages of symbol table space allocated to hold 515 non-local and 0 local symbols.

370 source lines were read in Pass 1, producing 22 object records in Pass 2.

11 pages of virtual memory were used to define 10 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name

Macros defined

_S255\$DUA28:[SYSLIB]STARLET.MLB;2

7

562 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:MTHCVTDG/OBJ=OBJ\$:MTHCVTDG MSRC\$:MTHCVTDG/UPDATE=(ENH\$:MTHCVTDG)

0258

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY